

Chapter 9: Using Kerberos

This chapter provides the information you need in order to manage your Kerberos tickets and work in a Kerberized environment. In particular, we cover ticket options and management, account access files and /root principal tickets. The Kerberos commands and features of Kerberized network programs are documented in Chapter 12: *Kerberos Command Descriptions* and Chapter 13: *Network Programs Available on Kerberized Machines*, respectively.

9.1 Ticket Properties and Options

Kerberos uses encrypted records called *tickets* to authenticate to Kerberized services (the terms *tickets* and *credentials* are used interchangeably). Tickets reside in a file called a ticket cache or credentials cache. Generally the only ticket you need to know about is the ticket-granting-ticket (TGT), which you obtain upon authentication to Kerberos. Kerberos tickets can be forwardable, renewable, post-dated and/or proxiable. The Kerberized versions of network programs generally provide options to exploit these features (see Chapter 13: *Network Programs Available on Kerberized Machines*).

Forwardable	A forwardable ticket can be “passed on” to a remote host, thereby allowing the user to connect to the host without further authentication. Generally only the TGT is set forwardable, since it can be used to obtain other needed tickets.
Renewable	A renewable ticket can have its lifetime extended, by action of the user, beyond the initial lifetime, up to an established limit (seven days at Fermilab).
Post-dated	A post-dated ticket becomes valid at a specified time in the future.
Proxiable	A proxiable ticket is like a forwardable ticket, except that the new ticket with the new address list is not allowed to be a TGT, it must be for some other service.



Our Kerberos implementation is integrated with AFS. This means that if your machine is part of the strengthened realm and it runs AFS, then when you obtain Kerberos credentials (or forward them to an AFS system), you also automatically get an AFS token. The other operations described in this chapter (e.g., listing, destroying tickets) also run on both the Kerberos tickets and the AFS token. The lifetime of the AFS token is set to the renewable lifetime of the Kerberos TGT.¹ (Note that if you're editing a file when the AFS token expires, it will suddenly become write-protected!)

9.1.1 Default Ticket Flags and Lifetimes

At Fermilab, the maximum ticket lifetime is set to 26 hours, and the default ticket lifetime as set on individual systems is constrained to be this value or less. The default flags and lifetimes of tickets obtained on a UNIX machine by login and **kinit** are set by entries in that machine's `/etc/krb5.conf`. (For other operating systems, the default values are typically set via a more user-friendly interface.) The maximum renewable ticket lifetime is seven days. We discuss the `krb5.conf` file in Chapter 16: *The Kerberos Configuration File: krb5.conf*.

9.1.2 Credential Caches

A *credential cache* is a file containing your tickets and session keys. Each window on your desktop that is running a remote session has a separate credential cache², with a separate expiration. The variable `$KRB5CCNAME` points to the credential cache in use on each host.³ Note that forwarded tickets and tickets obtained via **kinit** are stored in different caches.

9.1.3 Tickets for Root Instance of Kerberos Principal

The system administrator of a strengthened machine may require that authorized users obtain a `<username>/root` instance of their Kerberos principal in order to access the root account (and/or other sensitive accounts) on the machine. This is described in section 9.4.1 *What is a Root Instance of a Principal?* The `/root` instance has the properties of disallowing forwardable tickets and having a shorter default ticket lifetime.

1. Because AFS uses the Kerberos V4 ticket format, which squeezes the ticket lifetime into a small field, the expiration time of the AFS token may not *exactly* coincide with the end of the Kerberos ticket's renewable lifetime.

2. In some cases, there may be more than one per window.

3. Tickets generated by **kinit** end up in `/tmp/krb5cc_[UID]`, forwarded tickets go to `/tmp/krb5cc_p[PID]`, and hardware token tickets go into `/tmp/krb5cc_[ttyname]`.

9.2 Ticket Management

9.2.1 Obtaining Tickets (Authenticating to Kerberos)

The way to authenticate depends on your operating system and software. Upon authentication you get a Kerberos ticket-granting-ticket (TGT). As you access Kerberized services in the strengthened realm, the tickets needed for the services are granted automatically. As regular practice, authenticate locally and forward tickets to remote machines.

As of Kerberos v1_5, the **kinit** program is equipped with a warning that appears if the userid issuing the command doesn't own the console device. It is designed to help users avoid typing their password inadvertently over then network.

To authenticate:

UNIX desktop with Kerberos software and Kerberos login program	Log in, and provide your Kerberos password. See section 4.1 <i>Logging In at the Console of a Kerberized UNIX Machine</i> .
UNIX desktop with Kerberos and standard UNIX login program	Log in with your UNIX password, then run kinit . See section 4.1 <i>Logging In at the Console of a Kerberized UNIX Machine</i> . Also see 12.1 <i>kinit</i> .
Windows desktop with WRQ®	Navigate to START > PROGRAMS > REFLECTION > UTILITIES > KERBEROS MANAGER to open the Reflection Kerberos Manager application. With your principal tab selected, click AUTHENTICATE . Provide your Kerberos password as prompted (and click FORWARDABLE). See section 4.6 <i>Logging In Through WRQ® Reflection Software from Windows</i> .
Windows desktop with Leash32 and Kerberos	Using the Leash32 utility, navigate to START > PROGRAMS > KERBEROS UTILITIES > LEASH32 . Select GET TICKET on the ACTION menu. Provide your Kerberos password as prompted. See section 21.4 <i>Getting a Ticket</i> .
Macintosh desktop with Kerberos	For OS X, see section 23.1.4 <i>Authenticate to Kerberos</i> . For OS 9 and earlier: Invoke the KERBEROS CONTROL PANEL (from CONTROL PANELS under the Apple menu, from the KERBEROS MENU in the menu bar, or from the KERBEROS CONTROL STRIP module). Click GET TICKETS . Enter your Kerberos password on the pop-up screen. See section 7.2 <i>Logging In from a Macintosh</i> .
Remote UNIX host (from desktop with no Kerberos software installed)	Start an ssh (or telnet or FTP) session to a Kerberized host, use your CRYPTOCARD to generate a password, and log into the remote host using that one-time password. See section 4.4 <i>Connecting from a NonKerberized Machine: Portal Mode</i> .



When you're logging in as *root* you have to make sure you have tickets as some principal known to the KDC in order to access Kerberos network services. Whether you logged in as yourself and ran **ksu** to *root*, or logged in as *<yourprincipal>/root* over the network, you have credentials for the principal under which you previously authenticated.



If you have a laptop that you move from one network to another, then you will have to reobtain your credentials when you move to a new network because the IP address changes. Similarly, if you use DHCP, every time your IP address changes you need to get new credentials.

9.2.2 Viewing Tickets

The way to view your tickets depends on your operating system and software. Valid and expired tickets alike will be displayed.

To view tickets:

UNIX desktop with Kerberos software	Run the klist command (-f option recommended to show ticket flags). See section 12.2 <i>klist</i> .
Windows desktop with WRQ®	Navigate to START > PROGRAMS > REFLECTION > UTILITIES > KERBEROS MANAGER to open the Reflection Kerberos Manager application. Ticket should be visible on this window. Right-click on ticket to see ticket properties. See section 4.6 <i>Logging In Through WRQ® Reflection Software from Windows</i> .
Windows desktop with Leash32.	Using the Leash32 utility, navigate to START > PROGRAMS > KERBEROS UTILITIES > LEASH32 . Ticket should be visible on this window. See section 21.4 <i>Getting a Ticket</i> .
Macintosh desktop with Kerberos	For OS X, use the Unix method, or click on the ticket in the GUI. For OS 9 and earlier: Invoke the KERBEROS CONTROL PANEL (from CONTROL PANELS under the Apple menu, from the KERBEROS MENU in the menu bar, or from the KERBEROS CONTROL STRIP module). Ticket should be visible on this window. See section 7.2 <i>Logging In from a Macintosh</i> .
Remote Kerberized UNIX host	Run the klist command (-f option recommended to show ticket flags). See section 12.2 <i>klist</i> .

About the klist Command

The command **klist** displays your tickets (the **-f** option displays the flags set for the tickets), e.g.,:

```
% klist -f
```

This produces output of the form:

```
Ticket cache: /tmp/krb5cc_6302
Default principal: aheavey@FNAL.GOV
```

```
Valid starting      Expires      Service principal
12/08/99           11:29:47      12/09/99          00:29:47
krbtgt/FNAL.GOV@FNAL.GOV
Flags: FIA
12/08/99 11:29:48  12/09/99 00:29:47  afs/fnal.gov@FNAL.GOV
Flags: FA
```

- The first listed ticket is a Kerberos TGT (krbtgt) for the service principal `krbtgt/FNAL.GOV@FNAL.GOV`¹. Underneath it the flags are listed. This ticket has flags set for “forwardable”, “initial”, and “preauthenticated”.
- The second listed ticket indicates that AFS is running on this machine and that an AFS token has also been granted; this is again followed by a list of the flags associated with the ticket.

If you have no tickets you will see output like this:

```
klist: No credentials cache file found (ticket cache /tmp/krb5cc_6302)
```

Several options are available for **klist**, as listed in section 12.2 *klist* and in the man pages.

9.2.3 Destroying Tickets

Tickets can outlive an interactive session and they can be stolen. They are just encrypted records in a file. Therefore it’s a good idea to explicitly destroy your tickets when you log out. Similarly, if you are going to be away from your machine but don’t want to log out, it is safest to either destroy your tickets, or use a screensaver that locks the keyboard.

To destroy tickets:



UNIX desktop with Kerberos software

Run the **kdestroy** command. This destroys all the tickets in the cache to which `$KRB5CCNAME` points. To automate this, add the command **kdestroy** to your `.logout` file. See section 12.4 *kdestroy* or the man pages for a description of **kdestroy**.
If you’re sharing a credentials cache among several login sessions (by setting the `$KRB5CCNAME` variable), issuing the **kdestroy** command on any of the sessions destroys the tickets for all of them.

1. See *principal* in the glossary for an explanation of the syntax.

Windows desktop with WRQ®	Navigate to START > PROGRAMS > REFLECTION > UTILITIES > KERBEROS MANAGER to open the Reflection Kerberos Manager application. Tickets should be visible on this window. Click on CLEAR TICKETS . To automate the clearing of tickets, you can click CLEAR ALL TICKETS ON SHUTDOWN from the CONFIGURATION menu.
Windows desktop with Leash32	Using the Leash32 utility, navigate to START > PROGRAMS > KERBEROS UTILITIES > LEASH32 . Ticket should be visible on this window. Click on DESTROY TICKET(S) . To automate the clearing of tickets, you can click DESTROY TICKETS/TOKENS ON EXIT from the OPTIONS menu to clear tickets when you exit Leash32.
Macintosh desktop with Kerberos	For OS X, use the Unix method, or click Destroy Tickets on the GUI. For OS 9 and earlier: Invoke the KERBEROS CONTROL PANEL (from CONTROL PANELS under the Apple menu, from the KERBEROS MENU in the menu bar, or from the KERBEROS CONTROL STRIP module). Ticket should be visible on this window. Click on DESTROY TICKETS .
Remote Kerberized UNIX host	Run the kdestroy command.

Destroying Tickets Selectively

If you have several tickets in your cache and you run **kdestroy**, you'll destroy them all. But say you want to destroy only one or some of them. If your TGT is renewable, running **kinit -R** will discard all but the TGT, which gets renewed. If your tickets are forwardable, you can forward the TGT alone to your own machine by **rsh** or other program, and then overwrite your existing cache, e.g.,:

```
% rsh -F `hostname` cp \${KRB5CCNAME} ${KRB5CCNAME}
```

(Backquotes around *hostname*) If the KRB5CCNAME value has **FILE:** on the front of it (true of the recent kerberos releases), the preceding command will fail; in this case, try:

```
% rsh -F `hostname` cp `echo ${KRB5CCNAME} | sed -e sxFILE:xx` \
`echo ${KRB5CCNAME} | sed -e sxFILE:xx`
```

(All the quotes are backquotes.) To do anything more specific you'd have to write a program with the credentials cache API (which is beyond the scope of this document).

9.2.4 Forwarding Tickets

You can use your current, valid credentials on your desktop to get valid credentials on another machine by forwarding them.¹ You should forward tickets if you plan to use Kerberized services on the remote host (e.g., if you plan to connect from there to another remote Kerberized machine) and/or if you need an AFS token. To forward tickets, there are two steps:

- 1) you must first obtain a forwardable ticket,
- 2) and then make sure the “forward” option is used by your connection program.

The way to do this of course depends on your OS and software:

UNIX desktop with Kerberos software and Kerberos login program	To obtain a forwardable ticket, the <code>/etc/krb5.conf</code> must show <code>forwardable = true</code> for <code>login</code> under <code>[appdefaults]</code> . If not, check for <code>forwardable = true</code> for <code>kinit</code> . If this is true, run kinit . If false, run kinit -f . To forward your forwardable ticket to a remote UNIX host, use a Kerberized connection program with ticket forwarding on ^a (e.g., telnet -F).
UNIX desktop with Kerberos and standard UNIX login program	To obtain a forwardable ticket, check for <code>forwardable = true</code> for <code>kinit</code> in <code>/etc/krb5.conf</code> . If true, run kinit . If false, run kinit -f . To forward your forwardable ticket to a remote UNIX host, use a Kerberized connection program with ticket forwarding on (e.g., telnet -F). (See footnote a.)
Windows desktop with WRQ®	To obtain a forwardable ticket, click FORWARDABLE when you authenticate. See section 4.6 <i>Logging In Through WRQ® Reflection Software from Windows</i> . To forward your forwardable ticket to a remote telnet session, verify that the telnet configuration file you’re using specifies FORWARD TICKET on the SECURITY PROPERTIES window. See section 19.8 <i>Configuring WRQ® Reflection telnet Connections</i> . Note: WRQ®’s FTP client doesn’t support forwarding tickets. This only poses a problem for remote hosts running AFS since you don’t get your AFS token upon connection. See section 4.6.3 <i>Run an FTP Session to Kerberized Host</i> .

1. The KDC administrator has the option of disallowing forwardable tickets on a per-site or per-principal basis.

Windows desktop with Leash32, MIT Kerberos and Exceed 7	<p>To obtain a forwardable ticket, make sure your configuration specifies <code>Forwardable</code> under TICKET OPTIONS as described in section 21.3 <i>Configuring Kerberos using Leash32</i>.</p> <p>To forward your ticket to a telnet session, verify that the telnet configuration file you're using specifies <code>Forwarding</code> under KERBEROS 5 OPTIONS. See section 21.5 <i>Configuring the Exceed 7 Telnet Application</i>. Then run the telnet client.</p> <p>Note: The Exceed 7 FTP client cannot be Kerberized; try FileZilla FTP.</p>
Macintosh desktop with Kerberos	<p>For OS X, use the Unix method, or (GUI method not documented yet). For OS 9 and earlier: To obtain a forwardable ticket, edit your Preferences and check FORWARDABLE TICKETS ALWAYS.</p> <p>To forward the ticket via a BetterTelnet connection, check KERBEROS FORWARDING when you're configuring the Security portion of Favorites for that application.</p>
Remote Kerberized Host via Portal Mode	<p>When you obtain your ticket upon CRYPTOCARD login to a remote host, the ticket's properties are determined by the <code>/etc/krb5.conf</code> file on the host. Run klist -f to see if the <code>F</code> flag shows up indicating a forwardable ticket. If it doesn't, and if you used ssh to connect thus providing an encrypted connection, then you can run kinit -f to get one, BUT ONLY RARELY!</p> <p>To forward your forwardable ticket to a remote UNIX host, use a Kerberized connection program with ticket forwarding on.</p>

a. Check for `forward = true` in `[appdefaults]` section of `/etc/krb5.conf` for your program of choice (ssh has its own configuration). If false, use the program's command line option for ticket forwarding; these are documented in Chapter 13: *Network Programs Available on Kerberized Machines*.

Descriptions of the forwarding option (and other Kerberos functions) added to the connection programs in the Kerberos V5 package can be found in Chapter 13: *Network Programs Available on Kerberized Machines* and at <http://hoth.stsci.edu/public/krb5/user-guide.html#SEC16>.

Tickets and IP Addresses: How forwarding works

A ticket normally includes a list of IP addresses from which it may be used. A forwardable ticket may be presented to the KDC to obtain a ticket with a different address list, which can then be forwarded to another host and used from there.

The IP address (or list of IP addresses) of the client is encoded inside of every Kerberos ticket. This information is used by application servers and the KDC to verify the address of the client. By default, then, a ticket that was acquired

on one host cannot be used on another. This is where forwarding comes in. A forwardable ticket (usually a TGT) can be used to request a new ticket, but with a different IP address.



The new IP addresses to be included in a forwarded ticket are determined from the DNS entry for the target hostname. If that host turns out to have other IP addresses which are not listed under that name, the forwarded ticket may or may not be usable, depending on how that host routes packets to the KDC or to the other nodes you try to access.



A Note about AFS tokens and Forwarding

Telnet, **rsh** and **rcp** and **ftp** work without strictly requiring that credentials be forwarded. These programs always present a service-specific credential to get access, but don't necessarily forward it to the remote system.

- For **telnet**, you typically want to forward your credential (and automatically obtain an AFS token as needed), in order to avoid running **kinit** over the network. But if you don't plan to make any further connections from the remote host, and AFS is not running, forwarding is not strictly necessary.
- For **rsh** you'd only need to forward if the remote process you're invoking might need to make a further network access, or access files in an AFS file system.
- For **rcp** and **ftp** only the AFS case would lead you to want to forward credentials.

A Word about Ticket Caches and Forwarding

Forwarding actually involves asking the KDC to rewrite the ticket to be valid from the remote machine instead of from your desktop. In the case of telnet, the telnetd on the remote host receives the forwarded ticket, creates a credential cache file in `/tmp` and puts its name into the variable `$KRB5CCNAME`. The shell spawned by telnetd inherits this variable, so any kerberos client programs you run in that shell will use the forwarded ticket in that cache. If you then start an xterm process, it and the shell (or other process) it spawns inherit this environment variable and therefore know where to find your ticket. When the shell process created by telnetd exits, telnetd destroys the credential cache it created -- unless the host's `/etc/krb5.conf` tells telnetd "retain_ccache = true". As a user, you have no control over that setting.

Example (UNIX)

You will automatically obtain a forwardable ticket if under [appdefaults] in /etc/krb5.conf you see forward=true set for kinit or login, depending on how you got your ticket. You can always run **klist -f** and look for the **F** flag in the output if you're not sure:

```
12/08/99 11:29:47 12/09/99 00:29:47 krbtgt/FNAL.GOV@FNAL.GOV
Flags: FIA
```

If you need to replace your ticket with a forwardable one, run **kinit -f**.

Now, to forward this ticket to a remote host via telnet, first check under [appdefaults] in /etc/krb5.conf to see if forward=true is set for telnet. If so, just run **telnet <host>**. If not, run **telnet -f <host>** or **telnet -F <host>**. With **-f**, the forwarded ticket on the remote host is not set as reforwardable, and thus you can't forward it from that host to another. With **-F**, the forwarded ticket is marked as reforwardable from that host.

9.2.5 Renewing Tickets

In order to support both long interactive sessions and batch jobs, tickets can be issued as *renewable*¹, and given a *renewable lifetime*. This lifetime must be less than or equal to the maximum allowable renewable lifetime, which is set to seven days at Fermilab. A renewable ticket still has the normal lifespan (up to 26 hours), but before it expires it can be renewed as long as its renewable life has not expired. Once the ticket expires, new connections cannot be opened, but existing connections are not terminated. The lifetime of the AFS token that you get is equal to the Kerberos ticket's renewable lifetime.

1. If the /etc/krb5.conf file on the machine sets renewable=true and default_lifetime=<value greater than 26 hours>, the user will get a renewable ticket by default when they first log in. The Fermilab template for this file does not set renewable=true, but the system administrator can change this.

Make sure you read about **k5push** in section 9.2.6 *Update Tickets on Remote Terminal Sessions*, which renews tickets on multiple remote sessions simultaneously. For a local session, how you go about requesting a renewable ticket and renewing it depend upon your OS and software:

UNIX desktop with Kerberos software	To request a renewable ticket, use kinit -r <renewable_lifetime> . This requires password entry, therefore it must only be performed at the keyboard of a strengthened machine or (infrequently) over an encrypted connection. To renew the ticket, use kinit -R before the ticket expires. kinit -R does not require password entry.
Windows desktop with WRQ®	To request a renewable ticket, navigate to START > PROGRAMS > REFLECTION > UTILITIES > KERBEROS MANAGER to open the Reflection Kerberos Manager application. With your principal tab selected, click AUTHENTICATE . Provide a non-zero value for RENEWABLE DURATION . See section 4.6 <i>Logging In Through WRQ® Reflection Software from Windows</i> . To renew the ticket, again open the Reflection Kerberos Manager application. With your principal tab selected, click
Windows desktop with Leash32 and Kerberos	To request a renewable ticket, use the command prompt, and type kinit -r <renewable_lifetime> , as for UNIX. To renew the ticket, use the kinit -R option before the ticket expires. kinit -R does not require password entry.
Macintosh desktop with Kerberos	It appears that tickets obtained via the Macintosh Kerberos software are renewable by default (although the “R” flag does not appear). For OS X, use the Renew Tickets button on the GUI, or use the Unix method. For OS 9 and earlier: to renew a ticket, invoke the KERBEROS CONTROL PANEL (from CONTROL PANELS under the Apple menu, from the KERBEROS MENU in the menu bar, or from the KERBEROS CONTROL STRIP module). Click RENEW TICKETS...
Remote Kerberized host via Portal Mode	Run the command new-portal-ticket and use your CRYPTOCARD.

Example

Request a renewable ticket with a maximum renewable lifetime of four days using the **-r** option:

```
% kinit -r 4d
```

Password for aheavey@FNAL.GOV: <--- type your password here.

Then, before the default lifetime of 26 hours has passed (you cannot renew an expired ticket), and before four days expire, renew the ticket using the **-R** option:

```
% kinit -R
```

The ticket will remain active an additional 26 hours or until its original four day term expires, whichever comes first.

9.2.6 Update Tickets on Remote Terminal Sessions

What do you do when you have connections open to remote machines, and your tickets on these machines expire? Well, you most certainly *don't* run **kinit** over the network! And it turns out you don't have to exit and restart each session, either:

- You can push your valid tickets from your local machine to these remote machines via a script called **k5push**.
- From Windows (using **WRQ® Reflection**) you need to connect to a remote UNIX host first and run **k5push** from there, as we'll show you.
- For a session authenticated using a CRYPTOCard, use **new-portal-ticket**, as described in section 5.7 *Reauthenticate using your CRYPTOCard*.

k5push

Authenticate to Kerberos locally first before using **k5push**. The **k5push** script connects to an open session on a remote UNIX system using Kerberized **rsh**, and updates the remote ticket cache file in `/tmp` with the new tickets from your desktop machine. **k5push** does not create a ticket cache; one must already exist on the remote node. To run the script, type this command at your local session prompt:

```
% k5push <host1> [ <host2> <host3>...]
```

The script makes quite a few checks to make sure that the ticket file is really one of yours, and belongs to a running session. The **k5push** script is included in the Fermi Kerberos product as of v1_5. It is also available from http://www.fnal.gov/docs/strongauth/misc/k5push_script.txt.

k5push options

1. You can run this to an account with a different name:

```
% k5push <username>@<host> [ [<username>@]<host2> ...]
```

but be aware that if the target account is a shared account, you might update other users' ticket files with your tickets.

2. You can keep a list of systems to update in a text file, and run:

```
% k5push -f <file>
```

to update them simultaneously. (From UNIX, this file must be local; from Windows, this file must be on the UNIX host to which you connect.) The text file must list hosts and/or accounts on hosts each on a separate line, e.g.,:

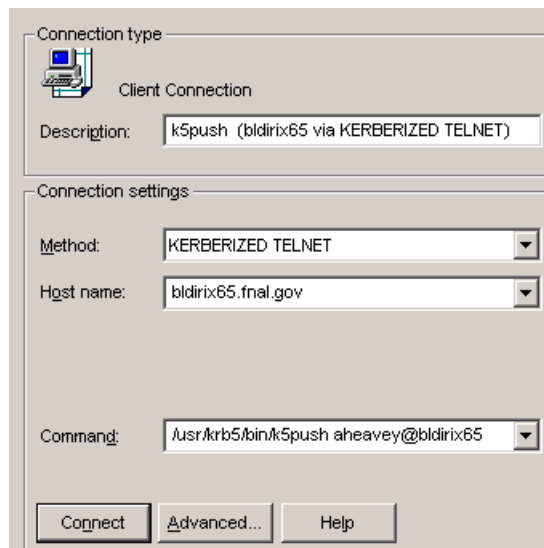
```
<host>.fnal.gov  
<host>.<domain>  
<account>@<host>.<domain>
```

Using k5push from Windows with WRQ®

As usual, use the **WRQ® Reflection Host - UNIX and Digital** program to run your remote VT100 sessions. Use the **WRQ® Reflection X Client Manager** to run the **k5push** command on a remote UNIX host session. If you use the **-f** option with a file, the file must exist on this UNIX host.

To update tickets on a single remote session:

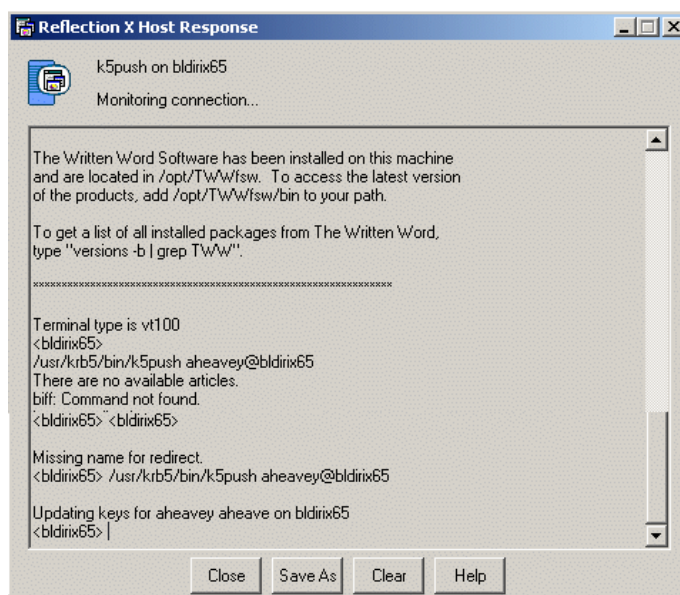
- verify that the **k5push** program exists on the remote UNIX host, and find its path (usually `/usr/krb5/bin`)
- invoke the **WRQ® Reflection X Client Manager**, if it's not already running
- on the right half of the **X Client Manager** window (as displayed in the default "Split Window Vertically" view), do the following:
 - add a description (e.g., `k5push (hostname via KERBERIZED TELNET)`)
 - select `KERBERIZED TELNET` as the connection method
 - enter the remote host name
 - enter the command `/path/to/k5push [username@]host`



- Still on the **X Client Manager** window, click **ADVANCED...**

- On **ADVANCED CLIENT CONNECTION SETTINGS**, make sure the prompt symbol you need is shown; also check **NEVER CLOSE CLIENT STARTER CONNECTION** (and if shown, check **HOST RESPONSE**).
- Click **CONFIGURE KERBEROS**.
- On **SECURITY PROPERTIES**, verify that **REFLECTION KERBEROS**, **MUTUAL AUTHENTICATION**, and **FORWARD TICKET** are checked. Verify that principal, realm and User ID are correct. Click **OK**.
- Back on **ADVANCED CLIENT CONNECTION SETTINGS**, click **OK**.
- Back on the **X CLIENT MANAGER** window, if necessary, open **CONNECTION > HOST RESPONSE** to monitor the process. Click **CONNECT**.

In the **HOST RESPONSE** window, you should see a session open to the remote host and see that it runs the **k5push** command as you entered it:



If you have multiple remote sessions and want to update credentials on all of them simultaneously:

- first choose one of your remote VT100 sessions as the “primary”
- on the primary host, verify that the **k5push** program exists, and find its path (usually `/usr/krb5/bin`)
- create a file on the primary host with all the necessary hostnames, as described above
- invoke the **WRQ® Reflection X Client Manager**, if it’s not already running
- fill in the right half of the **X Client Manager** window as described above, including the **ADVANCED...** options. Replace the command with:
`% /path/to/k5push -f filename`

9.3 Account Access by Multiple Users

Kerberos provides a way to grant account login access to multiple users, each with his/her own principal. There must be a `.k5login` file in the account's home directory and the principals must obtain credentials before logging into the shared account.

9.3.1 The `.k5login` File

The `.k5login` file is a text file that may exist in an account's home directory on a UNIX machine. It contains a list of the principals who have permission log into the account. Authenticated principals that are listed in the file can log in and use the account without limitations. A `.k5login` file is valid only on the individual strengthened host on which it resides.



Make sure that all principals that require login access are listed in it, **including your own FNAL.GOV principal!** Each principal must be on a separate line, with no trailing blanks.



This file overrides all other rules for granting login access!

Do you need a `.k5login` file?

As long as the only principal to log into your account is your own FNAL.GOV principal, and your principal matches your login id, you don't need a `.k5login` file. If other principals need login access to the account, and/or if your login id doesn't match your principal, you need one. And it must include your own principal!

Sample `.k5login`

```
xsmith@FNAL.GOV
qjones@FNAL.GOV
jenniferp@FNAL.GOV
jpedersen@MYUNIV.EDU
```

9.3.2 About Group Accounts

Sharing of any Kerberos password is a violation of Fermilab policy. Therefore, a multiple-user account must have a `.k5login` file in its home directory containing an entry for each user that needs to log into the account. The account may have but does not need a corresponding principal.



AFS ACLs should be set up so that everyone in the group can read (and write, if necessary) the files with his/her own AFS login and token. (This avoids the problem of running **klog** with a group AFS password.)

Users log in to the multiple-user account as follows:

- 1) Authenticate to Kerberos under your own account.
- 2) Log in to the multiple-user account, by identifying it on the connection program command line, and forward the ticket, e.g.,
`% telnet -f -l <group-account-name> <host>.`
- 3) Assuming tickets are automatically forwarded, you're now logged on under the account name, but your Kerberos ticket and AFS token are associated with your principal name.



- 4) Run **klog** to get an AFS token for the group account. If AFS is installed, you need to set the ACLs for file permissions for each principal.

9.3.3 The .k5users File

If you want to give restricted super user access to your account to another principal (access method limited to **ksu**; see section 12.5 *Kerberized su (ksu)*), you can create a `.k5users` file. The `.k5users` file is similar to the `.k5login` file, except that each principal is optionally followed by a list of commands which restricts the principal to those commands, and the file is only consulted by the **ksu** command.

Here is a sample `.k5users` file:

```
firstuser@MYUNIV.EDU /bin/ls /usr/bin/more
seconduser@MYUNIV.EDU /bin/ls
jenniferp@FNAL.GOV
jpedersen@MYUNIV.EDU
```

This restricts the first and second listed principals to the shown commands, and prohibits `jenniferp@FNAL.GOV` and `jpedersen@MYUNIV.EDU` from executing any command.



Two bombs:

- Be aware that arbitrary flags and arguments may be given to the listed commands by the authorized **ksu** user.
- If you list a principal more than once in this file, only the first entry is used.



If AFS is installed, you need to set the ACLs for file permissions for each principal.

9.4 Using Root Instance of your Principal

9.4.1 What is a Root Instance of a Principal?

A Kerberos principal has three parts and is of the form `primary/instance@REALM`. For a user, the instance portion is generally null, and the principal is of the form `primary@REALM`. If the instance is not null, the instance portion gives information that qualifies the primary, and is generally used to describe the intended use of the corresponding credentials. The root instance of a principal is also called a */root* principal. The word *root* in `<username>/root@FNAL.GOV` need not have anything to do with the UNIX *root* account, although that is presumed to be one of the most common uses. All */root* principals are created with the `DISALLOW_FORWARDABLE` flag set so that tickets are always unforwardable. The tickets also have a shorter default lifetime.

A root instance of your principal is only useful if your system administrator wants to make use of its restrictive ticket properties to protect sensitive accounts. Typically these accounts are set up with a `.k5login` file containing only */root* principals. Your system administrator should inform you if you need to obtain a */root* principal.

9.4.2 How do You Use your /root Principal?

To connect to such an account via a network connection from your desktop, you need to first **kinit** on your local machine as `<user>/root` (we use *me/root* as an example) and specify “nonforwardable ticket” with the **-F** flag¹:

```
% kinit -F me/root[@FNAL.GOV]
```

Now, connect to the sensitive account on the remote host using all of the options shown here:

```
% telnet -x -N -l <sensitive_account_name> <remote_host>
```

where:

- **-x** encrypts the connection (generally a good idea)
- **-N** tells the program not to forward tickets (you’ll get an error if you fail to include this)

1. If the Kerberos configuration file (`/etc/krb5.conf`) specifies forwarding “on” and you leave off the **-F**, you’ll get an error.

- **-l <sensitive_account_name>** logs you in directly to the named account. If you didn't include this, the remote host would try to log you into an account with the same name as your local UNIX username.

Note that once you're logged in remotely, you have no tickets. You cannot use any Kerberized services from here to connect to other accounts or machines.



If the sensitive account is in AFS space, or if you require read/write access to nonpublic AFS areas from that account, you need to authenticate the *machine* to AFS. Contact your AFS administrator for assistance.

9.4.3 How Should You NOT Use It?

There are some limitations associated with the use of */root* principals for access to privileged accounts, and that is why their use is not mandatory.

- You can't use **ksu** (or other Kerberized client) on a remote machine under your */root* principal because you don't have tickets on that machine.
- Never type your */root* principal password over the network except on rare, necessary occasions; always authenticate on your desktop machine.
- Do not use your */root* principal with unencrypted CRYPTOCARD connections, and rarely if at all with encrypted CRYPTOCARD connections. Remote authentication for the */root* principal would require transmission of the password.



9.4.4 How do you Maintain Credentials for your Normal Principal while Using the */root* Principal?

To maintain tickets on your desktop machine for both instances of your principal, you must keep the ticket caches separate. First authenticate under your normal principal, e.g.,:

```
% kinit [me[@FNAL.GOV]]
```

This gets you a ticket cache in the default area. You may find it useful to pick one of your local xterm windows to use for your */root* principal (maybe give it a special title bar or color) and set a separate ticket cache file there. In that window, reset the environment variable KRB5CCNAME to a location for the */root* principal ticket cache, then authenticate under your */root* principal to get (nonforwardable) tickets for this instance without overwriting the ones you got as "yourself":

```
% setenv KRB5CCNAME /tmp/krb5cc_me_root_$$
```

```
% kinit -F me/root[@FNAL.GOV]
```

When you request a Kerberized service, Kerberos will look at the credential cache to which KRB5CCNAME points, and assume that the principal holding this cache is the requestor. Reset this variable to the other cache as necessary.

